Homework 11: xv6 log

제출 마감: 2018. 12. 13. 11:00 am

제출처: 오준택 조교 (na94jun@gmail.com)

크래시 유발 예제

xv6 로깅의 목적은 파일시스템 작업 중 crash 발생시 문제가 되는 모든 디스크 업데이트 동작을 atomic하게 처리하는 것이다. 예를들면, 파일 생성에는 디렉토리에 새 항목을 추가하고 새 파일의 inode를 사용 중으로 표시하는 디스크 업데이트 작업이 포함된다. 만약 로깅을 하지 않을 경우, 디스크 업데이트 동작을 수행하는 전후에 발생한 충돌로 인해 파일 시스템이 잘못된 상태로 남게 된다.

아래의 몇 가지 예제는 파일을 부분적으로 만드는 방식으로 로깅 코드를 무효화시킨다.

첫 째, log.c 에 있는 commit() 함수를 아래와 같이 변경하시오.

```
#include "mmu.h"
#include "proc.h"
void
commit(void)
{
  int pid = myproc()->pid;
  if (log.lh.n > 0) {
    write_log();
    write_head();
    if(pid > 1)             // AAA
      log.lh.block[0] = 0; // BBB
    install_trans();
    if(pid > 1)             // AAA
      panic("commit mimicking crash"); // CCC
    log.lh.n = 0;
    write_head();
  }
}
```

BBB 라인은 log의 첫번째 블록을 0으로 쓴다. 해당 로깅 동작이 파일을 생성할 때 수행된 것이라면 로그의 첫 번째 블록은 type을 0이 아닌 값으로 업데이트한 새로운 file의 inode일 것이다. 라인 BBB는 업데이트된 inode을 0으로 작성되도록 하고 disk 내 inode는 여전히 할당되지 않은 상태로 남아 있게 한다. 라인 CCC는 충돌을 강제로 발생시킨다. 라인 AAA는 shell을 시작하기 전 파일을 생성하는 init process를 위한 동작에서 버그가 발생하는

것을 막는다. 즉, 위와 같이 commit()을 수정하면 CCC 라인을 수행할 때, 정전이 난 것과 같은 상태가 될 것이고, 첫번째 로그 블록에는 inode 블록이 아닌 boot 블록이 써질 것이다.

둘째, recover_from_log() 함수를 아래와 같이 변경하시오.

```
static void
recover_from_log(void)
{
  read_head();
  cprintf("recovery: n=%d but ignoring\n", log.lh.n);
  // install_trans();
  log.lh.n = 0;
  // write_head();
}
```

위와 같이 recover_from_log()를 수정하면 로그 복구(로그 영역에 커밋된 내용을 제자리에 쓰는 동작)가 동작하지 않는다.

만약, Makefile의 QEMUEXTRA 정의에서 -snapshot 옵션이 존재한다면 제거하시오.

이제 fs.img 를 제거하고 xv6를 실행하시오.

```
  % rm fs.img ; make qemu
```

“a” file을 shell 명령어를 이용해 생성하시오.

```
  $ echo hi > a
```

commit() 에서 발생하는 panic을 볼 수 있다. 지금까지 파일을 생성하는 중에 로깅이 되지 않는 시스템에서 충돌이 발생한 경우를 모사했다.

이제 fs.img를 유지하고 xv6를 재시작하시오.

```
  $ make qemu
```

그리고 a file을 다음 명령어를 통해 오픈하시오.

```
  $ cat a
```

type이 없다는 panic이 보일 것이다. 다음 중 충돌 전에 디스크에 수정된 파일 중 어떤 것이 기록되고 어떤 것이 기록되지 않았을까?

문제의 원인 생각하기

recover_from_log ()를 하기의 코드로 원복해라.

```
static void
recover_from_log(void)
{
  read_head();
  cprintf("recovery: n=%d\n", log.lh.n);
  install_trans();
  log.lh.n = 0;
  write_head();
}
```

fs.img를 유지한채로 xv6를 실행해라. 그리고 a file을 읽어라.

```
  $ cat a
```

이번에는 충돌이 발생하지 않는다. 왜 이번에는 정상 동작하는지 생각해 보시오.

그리고, file을 "echo hi > a" 로 만들어도 파일이 비어있는 이유는 무엇인가?

충돌 유발 예제는 여기까지다. 이제 로깅이 다시 동작하도록 commit() 의 수정 부분(즉, AAA,BBB 라인 등)을 제거하고 fs.img도 제거하라.

Streamlining Commit (로깅의 최적화)

File system에서 inode 영역 중 33번 block을 업데이트 한다고 가정하자. File system 은 bp=bread(block 33) 을 호출해 buffer 내 데이터를 업데이트 할 것이다. commit() 내 write_log() 는 디스크 상 log 영역의 블록에 업데이트된 블록(데이터)를 쓸 것이다. 예를 들어, 업데이트된 블록(33번 블록)이 log영역의 3번 블록에 쓰여졌다. write_log() 이후, install_trans()를 수행한다. install_trans()는 체크포인트다. log영역에서 업데이트된 블록 (33번 블록)이 포함된 3번 블록을 읽어 메모리 내 버퍼로 복사한다. 이후 디스크에 존재하는 33번째 블록에 다시 작성한다.

그러나, install_trans() 에서 수정된 33번째 블록은 버퍼 캐시에 여전히 남아있는 것을 알 수 있다. 버퍼 캐시가 커밋동작이 수행되기 전에 33번째 블록이 제거되면 왜 문제가 되는지 한번 생각해보자.

수정된 33번째 블록은 이미 캐시에 있으므로 로그에서 33번째 블록을 읽을 필요가 없다는 것을 알 수 있다.

과제: log.c를 수정하여 install_trans()를 commit()에서 호출할 때 로그에서 불필요한 읽기 작업이 수행되지 않도록 해라.

변경 내용을 테스트하려면 xv6에서 파일을 생성하고 다시 시작한 다음 파일이 아직 있는지 확인하시오.

Submit: 수정한 log.c]

제출 양식: hw11_학번.c